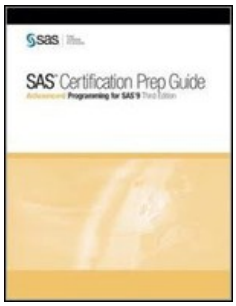

SAS Certification Prep Guide: Advanced Programming for SAS 9, Third Edition

by SAS Institute

SAS Institute. (c) 2011. Copying Prohibited.

Reprinted for Madhusmita Nayak, Accenture

madhusmita.nayak@accenture.com

Reprinted with permission as a subscription benefit of **Skillport**,
http://skillport.books24x7.com/

All rights reserved. Reproduction and/or distribution in whole or in part in electronic,paper or other forms without written permission is prohibited.

Chapter 8: Managing Processing Using PROC SQL

Overview

Introduction

The SQL procedure offers a variety of options that control processing. Some options control execution. For example, you can limit the number of rows read or written during a query. Other options control output. For example, you can control the appearance of long character columns, double-space output, or (as shown below) number your rows. Options are also available for testing and evaluating performance.

Row	FlightNumber	Destination
1	182	YYZ
2	219	LHR
3	387	CPH
4	622	FRA
5	821	LHR
6	132	YYZ
7	271	CDG
8	182	YYZ
9	219	LHR
10	387	CPH

Metadata is a description or definition of data or information. SAS session metadata is stored in Dictionary tables, which are special, read-only SAS tables that contain information about SAS libraries, SAS data sets, SAS macros, and external files that are available in the current SAS session. Dictionary tables also contain the settings for SAS system options and SAS titles and footnotes that are currently in effect. You can use the SQL procedure to access the metadata stored in Dictionary tables. For example, you can query a Dictionary table to find out which tables in a SAS library contain a specified column.

Objectives

In this chapter, you learn to

- use PROC SQL options to control execution
- use PROC SQL options to control output
- use PROC SQL to evaluate performance
- reset PROC SQL options without re-invoking the procedure
- use Dictionary tables and views to obtain information about SAS files.

Prerequisites

Before beginning this chapter, you should complete the following chapters:

- "Performing Queries Using PROC SQL" on page 4
- "Performing Advanced Queries Using PROC SQL" on page 29.

Specifying SQL Options

Remember that PROC SQL options are specified in the PROC SQL statement.

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

General form, PROC SQL statement:
PROC SQL *<option(s)>*;

where

option(s)
names the option(s) to be used.

Caution After you specify an option, it remains in effect until you change it or you re-invoke PROC SQL.

The following tables list the options for controlling processing that are covered in this chapter. A complete description and an example of each option appears in the following sections.

Table 8.1: Options to Control Execution

To do this…	Use this option…
Restrict the number of input rows	INOBS=
Restrict the number of output rows	OUTOBS=

Table 8.2: Options to Control Output

To do this…	Use this option…
Double-space the output	DOUBLE|NODOUBLE
Flow characters within a column	FLOW|NOFLOW|FLOW = *n*|FLOW = *n m*

Table 8.3: Options for Testing and Evaluating Performance

To do this…	Use this option…
Specify whether PROC SQL writes timing information for each statement to the SAS log	STIMER|NOSTIMER

Note For a complete list of options, see the SAS documentation for the SQL procedure.

Controlling Execution

Restricting Row Processing

When you are developing queries against large tables, you can reduce the amount of time that it takes for the queries to run by reducing the number of rows that PROC SQL processes. Subsetting the tables with WHERE clauses is one way to do this. Using the INOBS= and OUTOBS= options in PROC SQL is another way.

You already know that you can use the *OUTOBS*= option to restrict the number of rows that PROC SQL displays or writes to a table. However, the OUTOBS= option does not restrict the rows that are read. The *INOBS*= option restricts the number of rows that PROC SQL takes as input from any single source. The INOBS= option is similar to the SAS system option OBS= and is useful for debugging queries on large tables.

Note For more information about the OUTOBS= option, see "Performing Advanced Queries Using PROC SQL" on page 29.

Example

In the following PROC SQL set operation, INOBS=5 is specified. As indicated in the log, only five rows from each source table, *Sasuser.Mechanicslevell* and *Sasuser.Mechanicslevel2*, are used. The resulting table contains 10 rows.

```
proc sql inobs=5;
   select *
      from sasuser.mechanicslevel1
   outer union corr
   select *
      from sasuser.mechanicslevel2;
```


Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

Table 8.4: SAS Log

```
183   proc sql inobs=5;
184     select *
185        from sasuser.mechanicslevel1
186     outer union corr
187     select *
188        from sasuser.mechanicslevel2;

WARNING: Only 5 records were read from SASUSER.MECHANICSLEVEL1
         due to INOBS= option.
WARNING: Only 5 records were read from SASUSER.MECHANICSLEVEL2
         due to INOBS= option.
```

EmpID	JobCode	Salary
1400	ME1	$41.677
1403	ME1	$39,301
1120	ME1	$40.067
1121	ME1	$40,757
1412	ME1	$38,919
1653	ME2	$49,151
1782	ME2	$49,433
1244	ME2	$51,695
1065	ME2	$49,126
1129	ME2	$48,901

Tip You can use the PROMPT | NOPROMPT option with the INOBS= and OUTOBS= options so that you are prompted to stop or continue processing when the limits set by these options are reached.

Note For more information about PROC SQL set operations, see "Combining Tables Vertically Using PROC SQL" on page 132.

Caution In a simple query, there might be no apparent differences between using INOBS= or OUTOBS" Other times, it is important to choose the correct option. For example, using the average function on a column with the PROC SQL option INOBS=*10* returns an average of only the 10 values read for that column.

Controlling Output

Including a Column of Row Numbers

The *NUMBER*|*NONUMBER* option specifies whether the output from a query should include a column named ROW, which displays row numbers. NONUMBER is the default. The option is similar to the NOOBS option in the PRINT procedure.

Example

The following PROC SQL step specifies the NUMBER option. Output from the step includes a column named *Row*, which contains row numbers.

```
proc sql inobs=10 number;
   select flightnumber, destination
      from sasuser.internationalflights;
```

Row	FlightNumber	Destination
1	182	YYZ
2	219	LHR

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

3	387	CPH
4	622	FRA
5	821	LHR
6	132	YYZ
7	271	CDG
8	182	YYZ
9	219	LHR
10	387	CPH

Double-Spacing Output

In some cases, double-spacing your output can make it easier to read. The *DOUBLE*|*NODOUBLE* option specifies whether PROC SQL output is double-spaced.

The default is NODOUBLE.

Note The DOUBLE | NODOUBLE option does not affect the appearance of the HTML, PDF, or RTF output. To see the effect of this option, you must have text output selected in SAS Enterprise Guide.

Example

The following PROC SQL step specifies the DOUBLE option. The listing output from this step is double spaced. The HTML output from this step remains single-spaced.

```
proc sql inobs=10 double;
   select flightnumber, destination
      from sasuser.internationalflights;
```

The SAS System

FlightNumber	Destination
182	YYZ
219	LHR
387	CPH
622	FRA
821	LHR
132	YYZ
271	CDG
182	YYZ
219	LHR
387	CPH

Figure 8.1: Listing Output

Row	FlightNumber	Destination
1	182	YYZ

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

2	219	LHR
3	387	CPH
4	622	FRA
5	821	LHR
6	132	YYZ
7	271	CDG
8	182	YYZ
9	219	LHR
10	387	CPH

Figure 8.2: HTML Output

Flowing Characters within a Column

The *FLOW* | *NOFLOW* | *FLOW=n* | *FLOW=n m* option controls the appearance of wide character columns in listing output. The FLOW option causes text to be flowed in its column instead of wrapping the entire row. Specifying *n* sets the width of the flowed column. Specifying *n* and *m* floats the width of the column between limits to achieve a balanced layout.

Note The FLOW | NOFLOW | FLOW = *n*|FLOW =*nm* option does not affect the appearance of HTML, PDF, or RTF output. To see the effect of this option, you must have text output selected in SAS Enterprise Guide.

Example

The following PROC SQL step does *not* specify the FLOW option. Notice that in the output the name and values for the column *ZipCode* appear under the name and values for the column *FFID* due to the wide character columns.

```
proc sql inobs=5;
   select ffid, membertype, name, address, city,
          state, zipcode
      from sasuser.frequentflyers
      order by pointsused;
```

The SAS System

FFID ZipCode	MemberType	Name	Address	City	State
WD7152 72201	BRONZE	COOPER, LESLIE	66 DRIVING WAY	Little Rock	AR
WD8472 71655	BRONZE	LONG, RUSSELL	9813 SUMTER SQUARE	Monticello	AR
WD1576 72011	GOLD	BRYANT, ALTON	736 THISTLE DR.	Bauxite	AR
WD3947 72119	SILVER	NORRIS, DIANE	77 PARKWAY PLAZA	North Little Rock	AR
WD9347 72714	SILVER	PEARSON, BRYAN	9999 MARKUP MANOR	Bella Vista	AR

Figure 8.3: Output from PROC SQL Step without FLOW Option

Specifying **`flow=10 15`** causes the text within each character column to float between 10 and 15 spaces, which prevents the **`ZipCode`** column from wrapping underneath the **`FFID`** column.

```
proc sql inobs=5 flow=10 15;
   select ffid, membertype, name, address, city,
          state, zipcode
      from sasuser.frequentflyers
      order by pointsused;
```


Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

The SAS System

FFID	MemberType	Name	Address	City	State	ZipCode
WD7152	BRONZE	COOPER, LESLIE	66 DRIVING WAY	Little Rock	AR	72201
WD8472	BRONZE	LONG, RUSSELL	9813 SUMTER SQUARE	Monticello	AR	71655
WD1576	GOLD	BRYANT, ALTON	736 THISTLE DRIVE	Bauxite	AR	72011
WD3947	SILVER	NORRIS, DIANE	77 PARKWAY PLAZA	North Little Rock	AR	72119
WD9347	SILVER	PEARSON, BRYAN	9999 MARKUP MANOR	Bella Vista	AR	72714

Figure 8.4: Output from PROC SQL Step with FLOW Option

Testing and Evaluating Performance

Writing Timing Information for Each Statement

The PROC SQL option *STIMER* | *NOSTIMER* specifies whether PROC SQL writes timing information for each statement to the SAS log, instead of writing a cumulative value for the entire procedure. NOSTIMER is the default.

In order to use the STIMER option in PROC SQL, the *SAS system option* STIMER (the default) must also be in effect. Some host operating environments require that you specify the SAS system option STIMER when you invoke SAS. The STIMER system option controls the printing of performance statistics in the SAS log. If you use the system option alone, the results will contain timing information for the entire procedure, not on a statement-by-statement basis.

You can use the OPTIONS procedure to list the current settings of SAS system options. To find out if the SAS system STIMER option is enabled on your operating environment, submit the following program:

```
proc options option=stimer value;
run;
```

Table 8.5: SAS Log

```
Option Value Information For SAS Option STIMER
    Option Value: STIMER
    Option Scope: SAS Session
    How option value set: Shipped Default
```

Note PROC OPTIONS produces additional information that is specific to the operating environment under which you are running SAS. For more information about this and for descriptions of host-specific options, see the SAS documentation for your operating environment.

Example

Both of the queries in the following PROC SQL step list the name, address, city, state, and ZIP code of customers listed in the *Sasuser.FrequentFlyers* table. However, the second query lists only this information for customers who have earned more than 7000 points and used less than 3000 points.

When the PROC SQL statement is submitted without the STIMER option, timing information for both queries is written to the SAS log as a cumulative value for the entire procedure.

```
proc sql;
   select name, address, city, state, zipcode
      from sasuser.frequentflyers;
   select name, address, city, state, zipcode
      from sasuser.frequentflyers
      where pointsearned gt 7000 and pointsused lt 3000;
quit;
```

Note Timing information for a PROC SQL step is not written to the SAS log until a QUIT statement is submitted or another PROC or DATA step is started.

Table 8.6: SAS Log

```
28   proc sql;
```


Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

```
29   select name, address, city, state, zipcode
       from sasuser.frequentflyers;
30   select name, address, city, state, zipcode
       from sasuser.frequentflyers
31     where pointsearned gt 7000 and pointsused lt 3000;
32   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time 0.34 seconds
      cpu time 0.30 seconds
```

When the PROC SQL statement is submitted with the STIMER option, timing information is written to the SAS log for each SELECT statement.

```
proc sql stimer;
   select name, address, city, state, zipcode
      from sasuser.frequentflyers;
   select name, address, city, state, zipcode
      from sasuser.frequentflyers
      where pointsearned gt 7000 and pointsused lt 3000;
quit;
```

Table 8.7: SASLog

```
33   proc sql stimer;
NOTE: SQL Statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

34 select name, address, city, state, zipcode
     from sasuser.frequentflyers;
NOTE: SQL Statement used (Total process time):
      real time 0.22 seconds
      cpu time 0.17 seconds

35 select name, address, city, state, zipcode
     from sasuser.frequentflyers
36   where pointsearned gt 7000 and pointsused lt 3000;
NOTE: SQL Statement used (Total process time):
      real time 0.25 seconds
      cpu time 0.08 seconds

37 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time 0.29 seconds
      cpu time 0.03 seconds
```

Note When the STIMER option is used in PROC SQL, the exact wording of the Notes that are written to the SAS log might vary for different versions of SAS.

Note The STIMER option in PROC SQL is useful when an operation can be accomplished in more than one way and you are benchmarking each technique.

Although factors such as code readability and maintenance come into consideration, you might also want to know which PROC SQL step runs the fastest.

Resetting Options

Overview

After you specify an option, it remains in effect until you change it, or you re-invoke PROC SQL. You can use the RESET statement to add, drop, or change PROC SQL options without re-invoking the SQL procedure.

General form, RESET statement:

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

RESET *<option(s)>*;

where

option(s)

lists the options in any order.

Options are *additive*. For example, you can specify the NOPRINT option in a PROC SQL statement, submit a query, and submit the RESET statement with the NUMBER option, without affecting the NOPRINT option.

Example

Suppose you want to submit two PROC SQL queries in a single PROC SQL step. You want

- both queries to display only the first five rows of output
- the second query to display row numbers in the output.

In the following PROC SQL step, the PROC SQL statement specifies the OUTOBS= option to restrict the number of rows that will be displayed in the output. After the first SELECT statement, the RESET statement adds the NUMBER option to display row numbers in the result set.

```
proc sql outobs=5;
   select flightnumber, destination
      from sasuser.internationalflights;
reset number;
   select flightnumber, destination
      from sasuser.internationalflights
      where boarded gt 200;
```

The output, which contains two result sets, is shown below. The result set from the first SELECT statement reflects only by the OUTOBS= option. The result set from the second SELECT statement reflects both the OUTOBS= option and the NUMBER option that is specified in the RESET statement.

FlightNumber	Destination
182	YYZ
219	LHR
387	CPH
622	FRA
821	LHR

Row	FlightNumber	Destination
1	622	FRA
2	821	LHR
3	821	LHR
4	219	LHR
5	219	LHR

Now suppose you want to modify the PROC SQL step so that the result set from only the first SELECT statement is restricted to five rows of output. In the modified PROC SQL step, the OUTOBS= option is added to the RESET statement to change (reset) the OUTOBS= option that is specified in the PROC SQL statement. The modified step follows:

```
proc sql outobs=5;
   select flightnumber, destination
      from sasuser.internationalflights;
reset outobs= number;
```


Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

```
select flightnumber, destination
   from sasuser.internationalflights
   where boarded gt 200;
```

In the output, the result set from the second SELECT statement now contains *all* the rows that are generated by the query.

FlightNumber	Destination
182	YYZ
219	LHR
387	CPH
622	FRA
821	LHR

Row	FlightNumber	Destination
1	622	FRA
2	821	LHR
3	821	LHR
4	219	LHR
5	219	LHR
6	622	FRA
7	821	LHR
8	219	LHR
9	821	LHR
10	219	LHR
11	622	FRA
12	219	LHR
13	821	LHR
14	219	LHR
15	219	LHR
16	821	LHR
17	219	LHR
18	622	FRA
19	622	FRA
20	219	LHR
21	821	LHR
22	622	FRA
23	821	LHR

Using Dictionary Tables

Overview

Dictionary tables are commonly used to monitor and manage SAS sessions because the data is easier to manipulate than the output from procedures such as PROC DATASETS.

Dictionary tables are special, read-only SAS tables that contain information about SAS libraries, SAS macros, and external files that are in use or available in the current SAS session. Dictionary tables also contain the settings for SAS system

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

options and SAS titles and footnotes that are currently in effect. For example, the *Dictionary.Columns* table contains information (such as name, type, length, and format) about all columns in all tables that are known to the current SAS session.

Dictionary tables are

- created each time they are referenced in a SAS program
- updated automatically
- limited to read-only access.

Accessing a Dictionary table causes SAS to determine the current state of the SAS session and return the information that you want. Dictionary tables can be accessed by running a PROC SQL query against the table, using the *Dictionary* libref. Though SAS librefs are usually limited to eight characters, *Dictionary* is an automatically assigned, reserved word. You can also access a Dictionary table by referring to the PROC SQL view of the table that is stored in the *Sashelp* library.

The following table describes some of the Dictionary tables that are available and lists the corresponding *Sashelp* views. For a complete list of Dictionary tables, see the SAS documentation for the SQL procedure.

Dictionary table	Sashelp view	Contains
Catalogs	Vcatalg	information about catalog entries
Columns	Vcolumn	detailed information about variables and their attributes
Extfiles	Vextfl	currently assigned filerefs
Indexes	Vindex	information about indexes defined for data files
Macros	Vmacro	information about both user and system defined macro variables
Members	VmemberVsaccesVs catlgVslibVstableVst abvwVsview	general information about data library members
Options	Voption	current settings of SAS system options
Tables	Vtable	detailed information about data sets
Titles	Vtitle	text assigned to titles and footnotes
Views	Vview	general information about data views

Exploring and Using Dictionary Tables

You can query Dictionary tables the same way that you query any other table, including subsetting with a WHERE clause, ordering the results, creating tables, and creating PROC SQL views. Because Dictionary tables are read-only objects, you cannot insert rows or columns, alter column attributes, or add integrity constraints to them.

To see how each Dictionary table is defined, submit a DESCRIBE TABLE statement. The DESCRIBE TABLE statement writes a CREATE TABLE statement to the SAS log for the table specified in the DESCRIBE TABLE statement. After you know how a table is defined, you can use its column names in a subsetting WHERE clause in order to retrieve specific information.

Example

The *Dictionary.Tables* table contains detailed information about tables. The following DESCRIBE TABLE statement displays information about the *Dictionary.Tables* table in the log window. The information includes the names of the columns stored in the table.

```
proc sql;
   describe table dictionary.tables;
```

Table 8.8: SAS Log

```
create table DICTIONARY.TABLES
  (
```


Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

```
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   memlabel char(256) label='Dataset Label',
   typemem char(8) label='Dataset Type',
   crdate num format=DATETIME informat=DATETIME label='Date Created',
…);
```

To display information about the files in a specific library, specify the column names in a SELECT statement and the Dictionary table name in the FROM clause.

For example, the following PROC SQL step displays the columns

- **`Memname`** (name)
- **`Nobs`** (number of observations)
- **`Nvar`** (number of variables)
- **`Crdate`** (creation date) of the tables in the *Sasuser* library.

The Dictionary column names are specified in the SELECT statement and the Dictionary table name, *Dictionary.Tables*, is specified in the FROM clause. The library name, *Sasuser*, is specified in the WHERE clause.

Caution Note that you must specify the library name in the WHERE clause in uppercase letters (because that is how it is stored within SAS) and enclose it in quotation marks.

```
proc sql;
   select memname format=$20., nobs, nvar, crdate
      from dictionary.tables
      where libname='SASUSER';
```

Partial output is shown below.

Member Name	Number of Physical Observations	Number of Variables	Date Created
ACITES	50	4	05APR11:13:25:24
ADMIT	21	9	05APR11:13:25:17
ADMITJUNE	21	9	05APR11:13:25:17
AIRPORTS	50	4	05APR11:13:25:24
ALL	-	12	05APR11:13:25:25
ALLEMPS	50	5	05APR11:13:25:24
CAP2000	50	8	05APR11:13:25:24
CAP2001	50	8	05APR11:13:25:24
CAPACITY	50	7	05APR11:13:25:24
CAPINFO	50	7	05APR11:13:25:24

Note The **`nobs`** value for ALL is missing because it is a view, not a table.

Note Your output might differ from that shown above, depending on the contents of your *Sasuser* library.

You can also use Dictionary tables to determine more specific information such as which tables in a SAS library contain a specific column.

Example

The *Dictionary.Columns* table contains detailed information about variables and their attributes. As *m Dictionary.Tables*, the *Dictionary.Columns* table contains a column that is titled **`Memname`**, which lists the name of each table within a library.

```
proc sql;
   describe table dictionary.columns;
```


Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

Table 8.9: SAS Log

```
create table DICTIONARY.COLUMNS
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   name char(32) label='Column Name',
   type char(4) label='Column Type',
   length num label='Column Length',
…);
```

The following PROC SQL step lists all the tables in the *Sasuser* library that contain a column named **`EmpID.`** The dictionary column name, **`Memname`**, is specified in the SELECT statement. The Dictionary table, *Dictionary.Columns*, is specified in the FROM clause. The library name, *Sasuser*, and the column name, **`EmpID`**, are specified in the WHERE clause.

```
proc sql;
   select memname
      from dictionary.columns
      where libname='SASUSER'
            and name='EmpID';
```

Partial output is shown below.

Member Name
ALLEMPS
CONTRIB
ECONTRIB
EMPDATA
EMPDATU
EMPDATU2
FLIGHTATTENDANTS
FLIGHTSCHEDULE
MECHANICSLEVEL1
MECHANICSLEVEL2

Remember that you can also access a Dictionary table by referring to the PROC SQL view of the table that is stored in the *Sashelp* library. In the following PROC PRINT step, the *Sashelp* view *Vcolumn* is specified in the DATA= option. The results of the PROC PRINT step are identical to the preceding output.

```
proc print data=sashelp.vcolumn;
   var memname;
   where libname='SASUSER' and name='EmpID';
run;
```

Caution Note that column names in the WHERE clause must be specified in the same case that is used in the Dictionary table and must be enclosed in quotation marks.

Note You can use *Sashelp* views in any SAS procedure or DATA step. However, Dictionary tables can be only read by using the SQL procedure.

Additional Features

Restricting the Number of Loops

The LOOPS= option restricts the number of iterations of the inner loop in PROC SQL. By setting a limit, you can prevent queries from consuming excessive resources.

For example, joining three large tables without meeting the join-matching conditions could create a huge internal table that

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

would be inefficient to process. Use the LOOPS= option to prevent this from happening.

You can use the PROMPT | NOPROMPT option to modify the effect of the LOOPS= option so that you are prompted to stop or continue processing when the limit set by the LOOPS= option is reached.

Note You can use the number of iterations that are reported in the SQLOOPS macro variable (after each PROC SQL statement is executed) to gauge an appropriate value for the LOOPS= option. For more information about the SQLOOPS macro variable, see the SAS documentation for the SQL procedure.

Stopping Execution in PROC SQL after an Error

You already know that you can use the EXEC | NOEXEC option to specify whether a statement should be executed after its syntax is checked for accuracy. If the EXEC option is in effect, SAS checks the PROC SQL syntax for accuracy and, if no error is found, executes the SQL statement.

The ERRORSTOP | NOERRORSTOP option specifies whether PROC SQL stops executing if it encounters an error. This option is useful only when the EXEC option is in effect. The default is ERRORSTOP in batch or in a noninteractive session and NOERRORSTOP in an interactive SAS session.

ERRORSTOP instructs PROC SQL to stop executing the statements but to continue checking the syntax after it has encountered an error. ERRORSTOP has an effect only when SAS is running in batch or in noninteractive execution mode.

NOERRORSTOP instructs PROC SQL to execute the statements and to continue checking the syntax after an error occurs. NOERRORSTOP is useful if you want a batch job to continue executing SQL procedure statements after an error is encountered.

Summary

Contents

This section contains the following topics.

Text Summary

Specifying SQL Options

The SQL procedure offers a variety of options that affect processing. Some options control execution. For example, you can limit the number of rows read or written during a query or limit the number of internal loops PROC SQL performs. Other options control output. For example, you can flow character columns, number your rows, or doublespace output. Options are also available for testing and evaluating performance. Options are specified in the PROC SQL statement.

Restricting Row Processing

The OUTOBS= option restricts the number of rows that PROC SQL displays or writes to a table. The INOBS= option restricts the number of rows that PROC SQL takes as input from any single source. The INOBS= option is similar to the SAS system option OBS= and is useful for debugging queries on large tables.

Controlling Output

The NUMBER | NONUMBER option specifies whether the SELECT statement should include a column named ROW, which is the row number of the data as it is retrieved. NONUMBER is the default. The option is similar to the NOOBS option in the PRINT procedure.

In some cases, double-spacing your output can make it easier to read. The DOUBLE | NODOUBLE option specifies whether PROC SQL output is double-spaced in the listing output. The default is NODOUBLE.

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

The FLOW | NOFLOW | FLOW=*n*| FLOW=*n m* option controls the appearance of wide character columns in the listing output. The FLOW option causes text to be flowed in its column instead of wrapping the entire row. Specifying *n* sets the width of the flowed column. Specifying *n* and *m* floats the width of the column between limits to achieve a balanced layout.

Testing and Evaluating Performance

The STIMER | NOSTIMER option specifies whether PROC SQL writes timing information for each statement to the SAS log, in addition to writing a cumulative value for the entire procedure. NOSTIMER is the default. In order to use the STIMER option in PROC SQL, the SAS system option STIMER (the default) must also be in effect.

Resetting Options

After you specify an option, it remains in effect until you change it or you re-invoke PROC SQL. You can use the RESET statement to add, drop, or change PROC SQL options without re-invoking the SQL procedure.

Using Dictionary Tables

SAS session metadata is stored in Dictionary tables, which are special, read-only SAS tables that contain information about SAS libraries, SAS macros, and external files that are available in the current SAS session. A Dictionary table also contains the settings for SAS system options and SAS titles and footnotes that are currently in effect.

Accessing a Dictionary table causes PROC SQL to determine the current state of the SAS session and return the information that you want. Dictionary tables can be accessed by running a PROC SQL query against the table, using the *Dictionary* libref. You can also access a Dictionary table by referring to the PROC SQL view of the table that is stored in the *Sashelp* library.

To see how each Dictionary table is defined, submit a DESCRIBE TABLE statement. After you know how a table is defined, you can use its column names in a subsetting WHERE clause in order to retrieve specific information. To display information about the files in a specific library, specify the column names in a SELECT statement and the dictionary table name in the FROM clause. You can also use Dictionary tables to determine more specific information such as which tables in a SAS library contain a specific column.

Additional Features

The LOOPS= option restricts the number of iterations of the inner loop in PROC SQL. By setting a limit, you can prevent queries from consuming excessive resources.

The ERRORSTOP | NOERRORSTOP option specifies whether PROC SQL stops executing if it encounters an error.

Syntax

```
PROC SQL <option(s)>;
     DESCRIBE TABLE table-name <, …table-name>;
     SELECT column-l<, …column-n>
         FROM table-1 | view-l<, …table-n | view-n>
         <WHEREexpression>;
RESET <option(s)>;
QUIT;
```

Sample Programs

Querying a Table Using PROC SQL Options

```
proc sql outobs=5;
   select flightnumber, destination
      from sasuser.internationalflights;
reset number;
   select flightnumber, destination
      from sasuser.internationalflights
      where boarded gt 200;
quit;
```

Describing and Querying a Dictionary Table

```
proc sql;
   describe table dictionary.columns;
   select memname
```


Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

```
      from dictionary.columns
      where libname='SASUSER'
            and name='EmpID';
quit;
```

Points to Remember

- After you specify an option, it remains in effect until you change it or you re-invoke PROC SQL.
- The DOUBLE | NODOUBLE and the FLOW | NOFLOW | FLOW=*n*| FLOW=*n m* options do not affect the appearance of HTML, PDF, or RTF output that is created with the Output Delivery System.
- If you query a Dictionary table about the files in a specific library, the library name used in the WHERE clause must be specified in uppercase letters because that is how it is stored in SAS. Column names used in the WHERE clause must be specified in the same case as they appear in the Dictionary table.

Quiz

Select the best answer for each question. After completing the quiz, check your answers using the answer key in the appendix.

1. PROC SQL options are specified in ?

a. the PROC SQL statement.

b. an OPTIONS statement.

c. a SELECT statement.

d. the OPTIONS procedure.

2. Which of the following SQL options restricts the number of rows that PROC SQL takes as input from any single source? ?

a. OUTOBS=

b. INOBS=

c. OBS=

d. none of the above

3. Which PROC SQL step creates the output shown below? ?

EmpID	JobCode	LastName	FirstName
1124	FAI	FIELDS	DIANA
1422	FAI	FLETCHER	MARIE
1094	FAI	GOMEZ	ALAN
1113	FAI	JONES	LESLIE
1103	FAI	MCDANIEL	RONDA
1970	FA1	PARKER	ANNE
1132	FAI	PEARCE	CAROL
1116	FA1	RICHARDS	CASEY
1414	FAI	SANDERSON	NATHAN
1425	FA1	UNDERWOOD	JENNY
1130	FAI	WOOD	DEBORAH

Row	EmpID	JobCode	LastName	FirstName
1	1574	FA2	CAHILL	MARSHALL
2	1125	FA2	DUNLAP	DONNA
3	1475	FA2	EATON	ALICIA
4	1368	FA2	JEPSEN	RONALD

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

5	1411	FA2	JOHNSON	JACKSON
6	1441	FA2	LAWRENCE	KATHY
7	1477	FA2	MEYERS	PRESTON
8	1424	FA2	PATTERSON	RENEE
9	1413	FA2	PETERS	RANDALL
10	1555	FA2	RODRIGUEZ	JULIA

a.
```
a.  proc sql nonumber outobs=10;
      select *
         from sasuser.flightattendants
         where jobcode='FA1';
      select *
         from sasuser.flightattendants
         where jobcode='FA2';
```

b.
```
b.  proc sql number;
      select *
         from sasuser.flightattendants
         where jobcode='FA1';
    reset nonumber outobs=10;
      select *
         from sasuser.flightattendants
         where jobcode='FA2';
```

c.
```
c.  proc sql nonumber;
      select *
         from sasuser.flightattendants
         where jobcode='FA1';
    reset number outobs=10;
      select *
         from sasuser.flightattendants
         where jobcode='FA2';
```

d.
```
d.  proc sql;
      select *
         from sasuser.flightattendants
         where jobcode='FA1';
    reset outobs=10;
      select *
         from sasuser.flightattendants
         where jobcode='FA2';
```

4. Which of the following options does not affect the appearance of HTML, PDF, or RTF output? ?

a. NUMBER | NONUMBER

b. DOUBLE | NODOUBLE

c. FLOW | NOFLOW | FLOW=*n* | FLOW=*n m*

d. *b* and *c*

5. Which of the following statements is true regarding the STIMER option in PROC SQL? ?

a. The STIMER option in PROC SQL writes timing information for each statement to the SAS log.

b. The STIMER option in PROC SQL writes only cumulative timing information for the entire procedure to the SAS log.

c. When using the STIMER option in PROC SQL, the SAS system option STIMER must also be in effect.

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

d. *a* and *c*

6. A Dictionary table contains which of the following? ?

a. information about SAS libraries.

b. information about SAS data sets.

c. information about SAS macros.

d. all of the above

7. Dictionary tables are ?

a. created each time they are referenced in a SAS program.

b. updated automatically.

c. limited to read-only access.

d. all of the above

8. Dictionary tables can be accessed ?

a. by running a PROC SQL query against the table, using the *Dictionary* libref.

b. by referring to the PROC SQL view of the table that is stored in the *Sashelp* library.

c. by referring to the PROC SQL view of the table that is stored in the *Sasuser* library.

d. *a* and *b*

9. Which of the following PROC SQL steps displays information about the Dictionary table *Dictionary.Titles*? ?

a.
```
a. proc sql;
     describe dictionary.titles;
```

b.
```
b.  proc sql;
     describe table dictionary.titles;
```

c.
```
c.  proc sql describe table dictionary.titles;
```

d.
```
d.  proc sql describe dictionary titles;
```

10. Which of the following PROC SQL steps displays the name **(Memname)**, modification date **(Modate)**, number of variables **(Nvar)**, and the number of observations **(Nobs)** for each table in the *Sasuser* library? ?

a.
```
a.  proc sql;
     select memname, modate, nvar, nobs
        from dictionary.tables
        where libname='SASUSER';
```

b.
```
b.  proc sql;
     select memname, modate, nvar, nobs
        from dictionary.tables
        where libname='Sasuser';
```

c.
```
c.  proc sql;
     select memname, modate, nvar, nobs
        from 'SASUSER'
        where table=dictionary.tables;
```

d.
```
d.  proc sql;
     select SASUSER
        from dictionary.tables
```


Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

```
where cols= 'memname, modate, nvar, nobs';
```

Answers

1. Correct answer: a

   PROC SQL options are specified in the PROC SQL statement. After you specify an option, it remains in effect until you change it or you re-invoke PROC SQL.

2. Correct answer: b

   The INOBS= option restricts the number of rows that PROC SQL takes as input from any single source. The *IN0BS*= option is similar to the SAS system option OBS= and is useful for debugging queries on large tables. The OUTOBS= option restricts the number of rows that PROC SQL displays or writes to a table.

3. Correct answer: c

   After you specify an option, it remains in effect until you change it or you re-invoke PROC SQL. You can use the RESET statement to add, drop, or change PROC SQL options without re-invoking the SQL procedure. In the correct answer, the RESET statement adds the NUMBER option and the OUTOBS= option. The resulting output lists the first 10 rows in the table Sasuser.Flight-attendants where the value of **`Jobcode`** equals FA2 and includes a column named Row.

4. Correct answer: d

   The DOUBLE | NODOUBLE option specifies whether PROC SQL output is doublespaced in listing output. The FLOW | NOFLOW | FLOW-*n* | FLOW-*n m* option controls the appearance of wide character columns in listing output. Neither option affects the appearance of HTML output.

5. Correct answer: d

   The STIMER | NOSTIMER option in PROC SQL specifies whether PROC SQL writes timing information for each statement to the SAS log, instead of as a cumulative value for the entire procedure. NOSTIMER is the default. In order to use the STIMER option in PROC SQL, the SAS system option STIMER (the default) must also be in effect. If you use the system option alone, you will receive timing information for the entire procedure, not on a statement-by-statement basis.

6. Correct answer: d

   A Dictionary table is a special, read-only SAS data view that contains information about SAS data libraries, SAS data sets, SAS macros, and external files that are in use or available in the current SAS session. A Dictionary table also contains the settings for SAS system options that are currently in effect.

7. Correct answer: d

   Dictionary tables are created each time they are referenced in a SAS program, updated automatically, and limited to read-only access. Accessing a Dictionary table causes SAS to determine the current state of the SAS session and return the information that you want.

8. Correct answer: d

   Dictionary tables can be accessed by running a PROC SQL query against the table, using the Dictionary libref. Though SAS librefs are usually limited to eight characters, Dictionaiy is an automatically assigned, reserved word. You can also access a Dictionary table by refening to the PROC SQL view of the table that is stored in the Sashelp library.

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited

9. Correct answer: b

To see how a Dictionary table is defined, submit a DESCRIBE TABLE statement. The DESCRIBE TABLE statement writes a CREATE TABLE statement to the SAS log for the table specified in the DESCRIBE TABLE statement.

10. Correct answer: a

To display information about the files in a specific library, specify the column names in a SELECT statement and the Dictionary table name in the FROM clause. The library name in the WHERE clause must be specified in uppercase letters because that is how it is stored in SAS and it must be enclosed in quotation marks.

Reprinted for OET7P/639280, Accenture SAS Institute, SAS Institute Inc. (c) 2011, Copying Prohibited